Task Event Support for VxWorks

Emulate pSOS task event handling

Tue, Feb 10, 1998

The VME system code uses task event support provided by the pSOS kernel to a considerable degree.This note describes how to emulate that support as an aid to porting the system code to the VxWorks kernel.

The key is to use a binary semaphore that each task creates (empty) during task initialization. Along with the ***semaphore id*** that is kept in global memory (so that other modules can find it) there is an ***events*** word and a ***mask*** word. The ***events*** word holds events that have been sent to the task via TrigTask. The ***mask*** word holds events for which the task is waiting via its call to ExitTask.

A routine that desires to send an event calls TrigTask, which OR's in the event(s) that are to be sent into the task's ***events*** word. It also checks whether the task is waiting on an event by ANDing the ***events*** word with the ***mask*** word. If the result is zero, nothing more is done, because the events sent are not being awaited. If the result is nonzero, the task is waiting, so it needs to be enabled to run. A SemGive call places the task into the ready queue. When the task resumes execution, inside the ExitTask routine, it clears the ***mask*** word, then returns the value of the ***events*** word to the caller, taking care to zero the corresponding events that it returns in the ***events*** word. (This can be done by an Exclusive OR instruction to memory.) The task uses the returned ***events*** word value to determine what processing it should do. Then it (probably) loops back to the ExitTask call to await the next event.

ExitTask processing should first check whether the ***events*** word masked against the ***mask*** word is nonzero. If it is, it merely returns, clearing those events in the ***events*** word that it returns. In this case, it would not even have to set the ***mask*** word at all. As long as the ***mask*** word is zero, the task is not waiting on events.